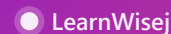


Esercizio di coding con l'IA

Costruisci il progetto AccessOps di questo modulo con ChatGPT o Claude partendo da un prompt già pronto che indirizza il modello alla specifica del modulo, poi rivedi, esegui ed estendi quello che ti restituisce.



Questo progetto non lo scriverai a mano. Lo dirigerai.

Quasi nessuno scrive più codice Wisej.NET da un file vuoto: un assistente scrive la prima bozza e lo sviluppatore decide se è giusta. Funziona però solo se conosci la materia, ed è quello che ti ha dato il resto di questo modulo. Qui affidi quelle conoscenze a ChatGPT o a Claude, ti fai restituire il progetto del modulo come soluzione Visual Studio funzionante, e lo rivedi come rivedresti la pull request di un collega.

Che cosa stai chiedendo

Aggiungi audit e gestione dei segreti ad AccessOps: un AuditService che registra approvazioni, modifiche ai permessi e download di documenti con utente, orario, indirizzo e correlation ID; una tabella di audit append-only; un correlation ID creato per ogni comando e scritto in ogni riga di log; stringhe di connessione e client secret OIDC spostati nella configurazione d'ambiente con una procedura di rotazione documentata; e una revisione dei log che oscura i campi sensibili.

Il modulo chiede questi deliverable, e il prompt qui sotto li chiede all'assistente esattamente così:

- ✓ AuditService e tabella di audit append-only
- ✓ Correlation ID in ogni comando e in ogni riga di log
- ✓ Segreti spostati nella configurazione d'ambiente
- ✓ Procedura di rotazione dei segreti
- ✓ Revisione dell'oscuramento nei log

Perché è fatto così

Ora AccessOps autentica le persone, ne risolve i permessi e protegge i propri documenti — e nulla di tutto questo aiuta la mattina in cui qualcuno chiede chi ha approvato la richiesta 4471, da quale indirizzo, e se qualcuno era stato rifiutato prima. I cinque moduli precedenti hanno deciso che cosa è consentito; questo decide che cosa è ricostruibile dopo. Chiude anche la falla che trasforma un piccolo incidente in uno grande: una password del database o un client secret OIDC in un file committato due anni fa.

Questi sono i termini che l'assistente deve azzeccare. Se quello che ti restituisce li usa con leggerezza, non ha capito il modulo:

Termine	Che cosa significa qui
audit trail	Un registro di business di ogni comando sensibile — chi ha agito, su che cosa, quando e da dove — conservato per un periodo dichiarato perché qualcuno dovrà risponderne.
archiviazione append-only	Una tabella di audit su cui l'account dell'applicazione può fare solo INSERT. Se il codice che scrive la cronologia può anche riscriverla, la cronologia non dimostra nulla.
evidenza di manomissione	Una sequenza senza buchi più una catena di hash, in cui l'hash di ogni riga copre i propri campi e l'hash precedente, così una riga modificata spezza la catena.
correlation ID	Un identificativo generato all'inizio di un comando e ripetuto su ogni riga di log e di audit che esso produce, così un singolo clic può essere ricostruito in seguito.
scope di logging	Un blocco di proprietà implicito — ILogger.BeginScope o un enricher di Serilog — che collega il correlation ID e l'autore a tutto ciò che viene registrato al suo interno.
archivio dei segreti	Il posto in cui una credenziale vive fuori dalla build: user secrets in locale, variabili d'ambiente in un container, un vault gestito in produzione — oppure una managed identity, senza nulla di memorizzato.
oscuramento	Una regola su che cosa non deve mai arrivare a un sink di log — password, token, cookie, stringhe di connessione e dati personali — fatta rispettare dal logging strutturato, non dalla memoria.

Costruiscilo — scegli il tuo assistente

Entrambi i pulsanti aprono una nuova scheda con lo **stesso prompt già pronto**. Il prompt indirizza il modello alla specifica di questo modulo — un unico file Markdown con la lezione, la traccia del lab e i deliverable — così lavora sul materiale vero invece di tirare a indovinare che cosa sia il Modulo 6. Puoi aprire tu stesso [module-06.md](#) per vedere esattamente che cosa leggerà.



Usa ChatGPT

Apri [chatgpt.com](#) con il prompt di questo modulo già caricato. Funziona con GPT-5 o Codex; qualsiasi piano che possa navigare il web recupererà la specifica.



Usa Claude

Apri [claude.ai](#) con lo stesso prompt già caricato. Chi usa Claude Code può invece incollarlo in un terminale dentro una cartella vuota.



Mandalo così com'è e ottieni un progetto, non un frammento

Il prompt si chiude chiedendo l'intera soluzione in un unico `.zip` con i passi per compilarla ed eseguirla. Scaricalo, apri la soluzione in Visual Studio, premi F5 e AccessOps si apre su `localhost`. Ti servono l'SDK .NET e il pacchetto NuGet `Wisej-4`; tutto il resto il prompt chiede all'assistente di scriverlo da sé.

Sto seguendo il Modulo 6 - "Audit, segreti e configurazione" - del corso LearnWisej "Pattern di sicurezza e autenticazione", e voglio costruire con te il progetto del modulo. Rispondimi in italiano.

Comincia leggendo la specifica del modulo. È un unico file Markdown (in inglese) che contiene l'intera lezione, la traccia del lab e i deliverable:

<https://learnwisej.com/assets/courses/security-and-auth-patterns/specs/module-06.md>

Poi costruisci un progetto Wisej.NET 4 completo che io possa aprire in Visual Studio:

1. Progetto "AccessOps", C#, net8.0 o successivo, pacchetto NuGet Wisej-4 4.1.x, host Kestrel che chiama `app.UseWisej()`.
2. Costruisci quello che chiede il lab della specifica: Aggiungi audit e gestione dei segreti ad AccessOps: un `AuditService` che registra approvazioni, modifiche ai permessi e download di documenti con utente, orario, indirizzo e correlation ID; una tabella di audit append-only; un correlation ID creato per ogni comando e scritto in ogni riga di log; stringhe di connessione e client secret OIDC spostati nella configurazione d'ambiente con una procedura di rotazione documentata; e una revisione dei log che oscura i campi sensibili.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - `AuditService` e tabella di audit append-only
 - Correlation ID in ogni comando e in ogni riga di log
 - Segreti spostati nella configurazione d'ambiente
 - Procedura di rotazione dei segreti
 - Revisione dell'oscuramento nei log
4. Segui la specifica ovunque nomini una classe, una proprietà o un evento di Wisej.NET - usa esattamente quello. Dove lascia una scelta aperta, dimmi quale API hai scelto e perché.
5. Tieni l'autorità sul server: stato e validazione vivono in .NET, il browser disegna e riferisce. Non spostare mai una decisione in JavaScript solo perché lì è più comodo.
6. Rendi il risultato dimostrabile: devo poterlo eseguire e vedere il comportamento di cui parla questo modulo, compreso che cosa succede quando qualcosa va storto.

Prima di scrivere codice, riformula con parole tue che cosa costruirai e dove starà ogni pezzo, così posso correggere il progetto finché costa poco.

Quando il codice è pronto, impacchetta l'intera soluzione in un unico file .zip scaricabile e dammi i passi esatti per compilarla ed eseguirla.

Infine rispondi a queste domande della specifica, usando come prova la tua implementazione:

- La tua tabella di audit è in sola scrittura per l'applicazione, e ogni riga è concatenata con gli hash. Indica una cosa che un attaccante con lo stesso login al database dell'applicazione può ancora fare al trail, e spiega esattamente come la mostrerebbe il passaggio di verifica.
- Un'approvazione parte dalla schermata di approvazione e termina in un task in background avviato con `Application.StartTask` dopo che l'utente si è disconnesso. Da dove prende quel task il correlation ID e l'autore, e che cosa si rompe se li legge invece dalla sessione?
- Uno sviluppatore rimuove il client secret OIDC dal file di configurazione tracciato e fa il push della modifica. Perché il segreto è ancora compromesso, e che cosa deve succedere prima che l'incidente sia davvero chiuso?

Chiedimi qualsiasi cosa non ti sia chiara prima di cominciare.

Poi rivedi quello che ti è tornato indietro

Codice che compila non è ancora codice giusto. Le domande interessanti sono domande di progetto: è questo il controllo Wisej.NET che il compito richiede davvero, segue il ciclo di vita che il framework ti offre, lo stato autorevole sta sul server o è stato parcheggiato nel browser perché era comodo? Leggi il risultato da revisore, non da cliente.

Questo modulo ti dà la checklist. Rispondi a ciascuna domanda guardando il codice che hai appena ricevuto:

- ✓ La tua tabella di audit è in sola scrittura per l'applicazione, e ogni riga è concatenata con gli hash. Indica una cosa che un attaccante con lo stesso login al database dell'applicazione può ancora fare al trail, e spiega esattamente come la mostrerebbe il passaggio di verifica.
- ✓ Un'approvazione parte dalla schermata di approvazione e termina in un task in background avviato con `Application.StartTask` dopo che l'utente si è disconnesso. Da dove prende quel task il correlation ID e l'autore, e che cosa si rompe se li legge invece dalla sessione?
- ✓ Uno sviluppatore rimuove il client secret OIDC dal file di configurazione tracciato e fa il push della modifica. Perché il segreto è ancora compromesso, e che cosa deve succedere prima che l'incidente sia davvero chiuso?

Tutto ciò che hai dovuto correggere vale la pena di scriverlo. Una breve nota con le convenzioni e le decisioni a cui sei arrivato, incollata nel primo messaggio della sessione successiva, vale più di qualsiasi prompt ingegnoso.



Per questo corso non c'è ancora un esempio di riferimento — il tuo è il primo

Il [repository degli esempi LearnWisej](#) contiene build già fatte per quasi tutti i corsi, ma non ancora per questo. Significa che non c'è niente con cui confrontarsi e nessuna scorciatoia: la specifica e la tua revisione sono tutta la rete di sicurezza. Tieni quello che costruisci — e se viene bene, ci farebbe piacere vederlo.

Tre prompt per andare oltre

Punta il tuo assistente sul progetto che hai appena costruito e prova questi. Ognuno è un esercizio di revisione ed estensione, non una riscrittura.

- ✓ **Dimostra che funziona davvero.** *"Passa in rassegna questo progetto deliverable per deliverable rispetto alla specifica del modulo su <https://learnwisej.com/assets/courses/security-and-auth-patterns/specs/module-06.md> e mostrami la prova che ciascuno è davvero fatto — il percorso nel codice, su che cosa cliccare e che cosa dovrei vedere. Dove un deliverable è soddisfatto solo in parte, dillo chiaramente invece di arrampicarti sugli specchi. Poi aggiungi quello che manca."*
- ✓ **Rivedilo come faresti con una pull request.** *"Accompagnami attraverso questo codice AccessOps come se stessi rivedendo la pull request di un collega e non l'avessi scritto tu. Dove vive lo stato, che cosa succede dall'inizio alla fine su una singola azione dell'utente, che cosa rimanderesti indietro per modifiche, e quali parti non ti sentiresti tranquillo a mettere davanti a un cliente? Sii specifico e cita le righe di cui parli — non riassumermi il progetto."*

- ✓ **Sostituisci un pezzo senza spostare le giunzioni.** *"Scegli la parte di questo progetto AccessOps che più evidentemente sta al posto di qualcosa di reale — una sorgente dati finta, un archivio in memoria, un servizio semplificato — e sostituiscila con un'implementazione di forma produttiva. Il contratto ai due lati non deve cambiare: stessi tipi, stesse chiamate, stesso comportamento in caso di errore. Dimmi che cosa l'implementazione reale ti ha costretto a gestire e che la versione provvisoria non gestiva."*



Portalo dentro il lab

Il lab pratico che chiude questo modulo chiede gli stessi deliverable che hai appena fatto produrre all'assistente. Usa quello che hai qui come forma della tua risposta e come prova a sostegno — non come la risposta in sé. Il lab viene valutato sul tuo ragionamento, e quella parte deve essere tua.